Санкт-Петербургский государственный университет

Кафедра математической теории игр и статистических решений

Хачатрян Альберт Гагикович

Выпускная квалификационная работа бакалавра

Анализ и прогнозирование безопасности дорожных маршрутов

Направление 010400

Прикладная математика и информатика

Научный руководитель,

кандидат физ.-мат. наук,

ассистент

Панкратова Ярославна Борисовна

Санкт-Петербург

2016

Содержание

Введение 3
Постановка задачи 5
Основные задачи 5
Математическая постановка задачи 5
Обзор литературы 7
Глава 1. Основные понятия 9
1.1. Понятие и принцип работы MVC (model view controller) . . 9
1.2. Взаимодействие с Google API 10
1.3. Статистический анализ 12
Глава 2. Метод анализа маршрута на примере 13
2.1. Анализ маршрута для водителя 13
2.2. Анализ маршрута для пешехода 16
Глава 3. Программная реализация 19
3.1. Сбор данных 19
3.2. Построение маршрута 20
3.3. Анализ построенного маршрута 21
Выводы 23
Заключение 24
Список литературы 25
Приложение 26

Введение

В данной работе исследуется проблема аварийности на дорогах общего пользования. Для исследования были собраны данные об аварийности на дорогах Санкт-Петербурга за 2015 год. При обработке данных была выдвинута основная цель работы, разработать общедоступный программный продукт направленный на повышение безопасности на дорогах.

Актуальность повышения уровня безопасности на дорогах продиктована тем, что дорожно транспортные происшествия (ДТП) являются частой причиной ущерба гражданам и угрозе их жизни и здоровью. Безопасность дорожного движения является одной из важных социально - экономических и демографических задач Российской Федерации.

Согласно данным государственной инспекции безопасности дорожного движения (ГИБДД), за 2015 год на территории Санкт-Петербурга и Ленинградской области произошло 7243 аварии. Из этого следует, что в прошлом году в среднем в день происходило около 20 аварий, что значительно меньше, чем показатели за 2014 год – 29 аварий. Однако получается, что практически каждый час на улицах Санкт-Петербурга происходит одна авария.

При обработке статистических данных о количестве и месте аварий в Санкт-Петербурге нами была поставлена задача придумать метод и реализовать общедоступный и понятный программный продукт-сервис, который строит маршрут как для водителей, так и для пешеходов, детально анализируя его и классифицируя как опасный или безопасный. Если существует несколько вариантов маршрутов, то сервис предложит наименее опасный из них. Также программа сообщит об особо опасных участках дороги, которые встретятся пользователю на его пути.

Таким образом, пользователь (водитель или пешеход) получает не только стандартную информацию, которую могут предложить современные карты или сервисы, но и детально разобранный по отрезкам (от перекрестка до перекрестка) маршрут с информацией об аварийности на каждом участке и классе всего маршрута (опасный или безопасный). В этом заключается новизна данной выпускной квалификационной работы.

При разработке метода были использованы данные с сайта ГИБДД [1]. Данный сервис опирается на уже существующие программные решения в этой области: был использован вспомогательный сервис Google API [5], а именно технология Google Maps Directions API [6]. С помощью данной технологии строится маршрут, при этом пользователь видит только обычную карту с проложенным маршрутом, в то время как программа, используя данную технологию, получает координаты перекрестков, длину каждого участка, название улицы, а также общие сведения о маршруте – загруженность дорог и общее время пути.

Используя полученные данные, программа проводит заложенные в нее вычисления, а именно считается аварийность маршрута на 1 км и риск попасть в аварию на этом маршруте; в качестве ответа выдает класс построенного маршрута и выделяет наиболее опасные участки дороги.

Постановка задачи

Основные задачи

Для решения выше описанной проблемы были собраны и обработаны данные об аварийности на территории Санкт-Петербурга за 2015 год. О каждой аварии известна следующая информация: день, когда произошла авария, район, в котором произошла авария, точный адрес ближайшего к месту аварии здания, вид аварии, количество машин, участвовавших в данной аварии, количество раненных и погибших.

Основные задачи исследования:

1. Осуществить анализ уровня безопасности построенного маршрута для водителя;
2. Осуществить анализ уровня безопасности построенного маршрута для пешехода;
3. Разработать общедоступный программный продукт, который позволяет анализировать любой маршрут построенный пользователем, с возможностью расширить сервис для всех регионов России.

Математическая постановка задачи

Пусть из пункта A в пункт B существует несколько маршрутов, назовем их $K_1, K_2, K_3, ..., K_n$.

Каждый маршрут состоит из нескольких участков - отрезков (от перекрестка до перекрестка), перекрестком будем считать смену одной улицы на другую. Обозначим участки маршрута следующим образом k_{ij}:

$$K_1 = (k_{11}, k_{12}, ..., k_{1m_i}), K_2 = (k_{21}, k_{22}, ..., k_{2m_i}), ..., K_n = (k_{n1}, k_{n2}, ..., k_{nm_i}),$$

где i принимает значения от 1 до n и отвечает за выбор определенного маршрута, j принимает значения от 1 до m_i и отвечает за выбор конкретного отрезка i-го маршрута.

Обозначим за l_{ij} длину каждого участка k_{ij}, измеряемую в километрах.

Введем переменную d, которая будет обозначать день недели, d может принимать значения от 0 до 6, 0 - понедельник, 1 - вторник и т.д.

Обозначим за A^d количество аварий, которое произошло в Санкт-Петербурге в d-ый день недели. И введем переменную a_{ij}^d - количество аварий, которые произошли в d-ый день недели на j-ом отрезке i-го маршрута.

Определение 1. Уровнем аварийности на участке k_{ij} в день d будем называть долю аварий, которая приходится на этот участок, в общем

числе аварий произошедших в Санкт-Петербурге в этот день. Обозначим аварийность буквой D_{ij}^d и будем считать по следующей формуле

$$D_{ij}^d = \frac{a_{ij}^d}{A^d}.$$

Таким образом, получаем долю аварий в d-ый день недели на j-ом отрезке i-го маршрута.

Определение 2. Под уровнем аварийности приходящейся на 1 км участка k_{ij} будем называть величину x_{ij}^d, которая считается следующим образом

$$x_{ij}^d = \frac{D_{ij}^d}{l_{ij}}.$$

Теперь для каждого маршрута можно составить выборку из случайных величин, которую будем обозначать X_i^d

$$X_i^d = (x_{i1}^d, x_{i2}^d, ..., x_{im_i}^d),$$

где i пробегает значения от 1 до n, d - от 0 до 6.

Определение 3. Под уровнем аварийности i-го маршрута будем называть выборочное среднее X_i^d, которое считается следующим образом

$$\bar{x}_i^d = \frac{1}{m_i} \sum_{j=1}^{m_i} x_{ij}^d.$$

Определение 4. Риском попасть в аварию будем называть выборочное среднее квадратическое отклонение, вычисленное для i-го маршрута. Обозначим как s_i^d и будем вычислять по следующей формуле

$$s_i^d = \sqrt{\frac{1}{m_i} \sum_{j=1}^{m_i} (x_{ij}^d - \bar{x}_i^d)^2}.$$

Опираясь на вычисления приведенные выше, а, именно, анализируя уровень аварийности маршрута и риск попасть в аварию, маршрут классифицируется как опасный или безопасный. Если маршрутов несколько, то пользователю предлагается наименее опасный из них.

Анализируя уровень аварийности на участке k_{ij}, выявляются самые аварийные участки дорог, и предлагается пользователю по возможности их объехать.

Обзор литературы

1. Официальный сайт ГИБДД.

   С помощью данного сайта были собраны все аварии за 2015 год, которые произошли на территории Санкт-Петербурга. Была изучена возможность получать данные с сайта в автоматизированном режиме, но на данный момент сайт ГИБДД не поддерживает технологию с помощью которой можно автоматизировать этот процесс.

2. Буре В. М., Парилина Е. М. Теория вероятностей и математическая статистика.

   Данная книга содержит основные разделы курса теории вероятностей: свойство вероятностной меры, формулы элементарных вероятностей, классическое определение вероятности, геометрическая вероятность, виды сходимостей случайных величин, характеристические функции и др. В данной дипломной работе была использована теория изложенная в гл. 5 (Случайные величины).

3. Орлов А.И., Математика случая: Вероятность и статистика – основные факты.

   В данной книге рассматриваются вероятностно-статистические основы современных статистических методов. Изложены основные понятия, которые используются при применении статистических методов. Особое внимание уделено непараметрическим подходам, статистике нечисловых данных и другим перспективным элементам высоких статистических технологий. Для данной работы использовался материал изложенный в гл. 5 (Выборочные характеристики).

4. Капский Д. В., Пегин П. А. Методика прогнозирования аварийности на регулируемом перекрестке.

   Данная статья была изучена с целью ознакомиться с уже существующими метода анализа маршрута. Метод предложенный в статье, не подходит под данные, которые используются в данной работе, т. к. авторы статьи учитывают направления движения автомобилей, время работы светофора, количество пешеход и машин, которые воспользовались перекрестком. И опираясь на эти показателей выявляются наиболее опасные участки перекрестка и строится прогноз.

   Такой метод не применим к данным, которые используются в данной работе.

5. Джефф Форсье, Пол Биссекс, Уэсли Чан, "Разработка веб - приложений".

В данной книге рассказывается о создании веб-приложений. Затронуты основные проблемы, которые возникают при их создании и изложены основные технологии, которые используются при создании веб-приложений. В книге основной уклон сделан в сторону языка Python, однако технологии и методы, которые там используются можно перенести практически на любой язык программирования. Сервис представленный в данной работе, собран по данной книге, особое внимание было уделено разделу 3 (Понимание моделей, представлений и шаблонов), разделу 4 (Преимущества ORM) и разделу 5 (Адреса URL, механизмы HTTP и представления).

6. Робин Никсон, "Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript".

   В данной книге описывается создание веб-сайтов от начало и до конца. Особое внимание уделяется взаимодействию веб-сайта с базой данных, так же рассмотрены технологии, которые применяются при создании динамических страниц, способных обновлять данные без перезагрузки самой страницы. Из данной книги были взяты основные принципы создания и взаимодействия с базами данных.

7. Бен Хеник, "HTML и CSS путь к совершенству".

   На основе данной книги были изучены основные принципы верстки веб-сайтов, способы и технологии визуального отображения данных. Были изучены технологии позволяющие придать сайту адаптивность, чтобы он корректно отображался на устройствах с разным разрешением экрана.

8. Официальный сайт документации Google Maps API.

   С помощью данной документации, были изучены основные способы взаимодействия с Google API. Были изучены возможности, которые предоставляет данный сервис и выбраны те, которые подходят для решения проблемы описанной в данной работе. С помощью технологии Google Maps Directions API строится маршрут в данной работе, с помощью этой технологии он разбивается на участки.

Глава 1. Основные понятия

1.1. Понятие и принцип работы MVC (model view controller)

MVC – это широко используемая техника разработки. Идея MVC заключается в разделении динамических приложений (как веб-приложений, так и других) на составляющие (модель–представление–контроллер). Это означает, что приложение делится на модель, управляющую данными, представления, определяющее, как будут отображаться данные, и контроллер, осуществляющий посреднические функции между первыми двумя уровнями и дающий пользователю возможность запрашивать данные и управлять ими.

Разделение приложения таким способом обеспечивает необходимую гибкость и позволяет многократно использовать один и тот же программный код. Например, существует модуль, способный отображать числовые данные в графическом виде, – этот модуль можно использовать для различных наборов данных при условии наличия связующего звена между модулем и данными.

На сегодня это самая популярная парадигма программирования, которая используется при веб-разработке. Сервис, представленный в данной работе, также использует MVC [4].

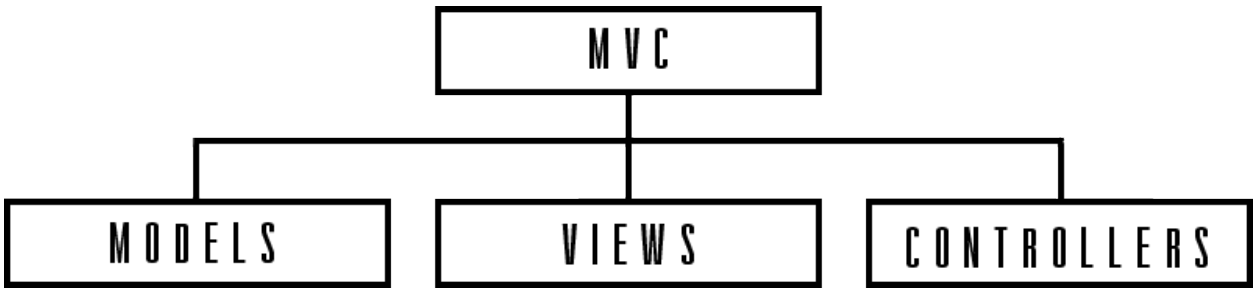

Рис. 1: Структура MVC

Ключевые принципы MVC:

1. Модели (models) - ответственны за данные приложения и доступ к базе данных.
2. Контроллеры (controllers) - отвечают за взаимодействие пользователя с системой. При необходимости контроллеры получают данные из моделей.
3. Представления (views) - выводят данные, полученные от контроллера.

Прямой связи между представлениями и моделями не существует. Наглядная демонстрация принципа работы MVC представлена в качестве блок схемы на рис. 2.

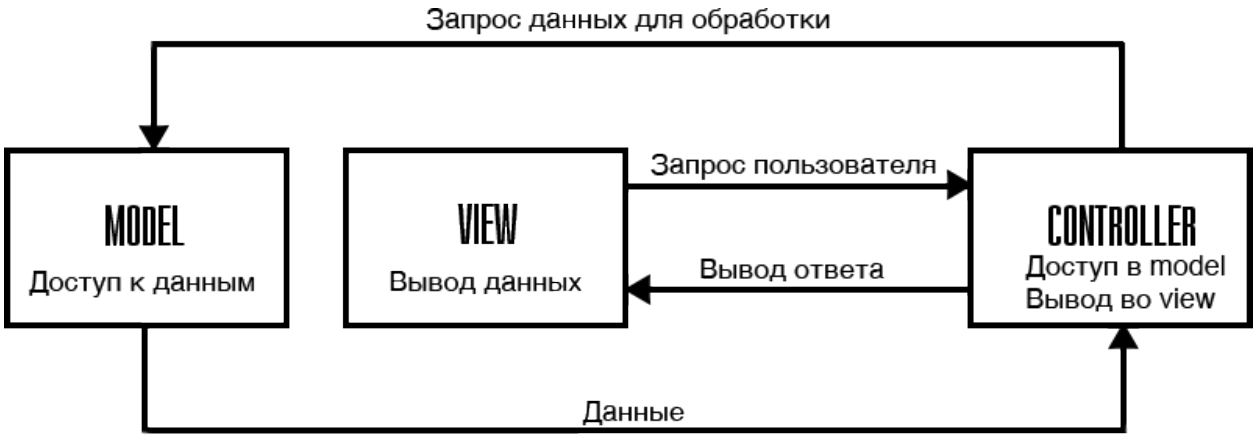

Рис. 2: Принцип работы MVC

Рассмотрим представленный принцип на примере данной работы. В качестве данных (models) у нас используется база аварий. В качестве контроллера используются методы и алгоритмы, представленные в данной работе, для запроса и обработки данных. В качестве визуальной составляющей (view) используется страница сайта. Таким образом, изначально пользователь видит страницу сайта, затем отправляет запрос с информацией, откуда и куда он хочет поехать. Контроллер получает эти данные, применяя к ним свои прописанные методы, происходит первый этап обработки, затем запрашивает из базы (models) аварии по маршруту, после получения аварий происходит вторая стадия обработки данных. Затем составляется ответ и отправляется во view и пользователь видит маршрут и информацию о нем.

1.2. Взаимодействие с Google API

API (application programming interface) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

Формат взаимодействия с Google API [5] может быть двух видов: json запрос или xml запрос. В данной работе будем использовать json формат взаимодействия, так как он работает быстрее. Запрос отправляемый на сервер Google API выглядит следующим образом:

https://maps.googleapis.com/maps/api/directions/json?origin=A
&destination=B&key=you-api-key

Где вместо A вписывается адрес, откуда нужно строить маршрут, вместо B - адрес конечной точки маршрута. Параметр key - это индивидуальный ключ, который выдается сервисом разработчику для взаимодействия с API. В данной работе также используется параметр mode, который

отвечает за построение маршрута для пешехода или водителя.

В ответ на запрос от сервера приходит следующая информация о маршруте: общая длинна пути, предполагаемое время в пути, далее идёт пошаговая инструкция с координатами перекрестков, с которых нужно будет совершить поворот, информации о длине каждого участка дороги, по которому проедет пользователь.

На рис. 3 для наглядности приведен фрагмент ответа сервера на запрос, где вместо A было вписано «Санкт-Петербург метро Нарвская», а в качестве конечной точки маршрута B - «Санкт-Петербург метро Елизаровская»


```
"copyrights" : "Картографические данные © 2016 Google",
"legs" : [
   {
      "distance" : {
         "text" : "11,0 км",
         "value" : 11010
      },
      "duration" : {
         "text" : "23 мин.",
         "value" : 1405
      },
      "end_address" : "Метро Елизаровская, Санкт-Петербург, Россия, 192029",
      "end_location" : {
         "lat" : 59.8960882,
         "lng" : 30.4227513
      },
      "start_address" : "Нарвская, Санкт-Петербург, Россия, 190020",
      "start_location" : {
         "lat" : 59.90109099999999,
         "lng" : 30.2742376
      },
      "steps" : [
         {
            "distance" : {
               "text" : "24 м",
               "value" : 24
            },
            "duration" : {
               "text" : "1 мин.",
               "value" : 5
            },
            "end_location" : {
               "lat" : 59.9012986,
               "lng" : 30.2741305
            },
            "html_instructions" : "Направляйтесь на \u003cb\u003eсевер
            "polyline" : {
               "points" : "ylrlJ_}wwDKF]L"
            },
            "start_location" : {
               "lat" : 59.90109099999999,
               "lng" : 30.2742376
            },
            "travel_mode" : "DRIVING"
         },
```


Рис. 3: Пример json ответа

1.3. Статистический анализ

Статистические данные – это результаты наблюдений (измерений, испытаний, опытов, анализов). Функции результатов наблюдений, используемые, в частности, для оценки параметров распределений или для проверки статистических гипотез, называют «статистиками». Если в вероятностной модели результаты наблюдений рассматриваются как случайные величины (или случайные элементы), то статистики, как функции случайных величин (элементов), сами являются случайными величинами (элементами). Для описания данных используются различные характеристики, мы будем использовать выборочные характеристики [2, 3].

В качестве выборочных средних величин будем использовать выборочное среднее арифметическое (сумму значений рассматриваемой величины, полученных по результатам испытания выборки, деленную на ее объем):

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

где n – объем выборки, x_i – результат измерения i-ого элемента выборки.

В качестве выборочных показателей рассеивания результатов наблюдений чаще всего используют выборочную дисперсию, выборочное среднее квадратическое отклонение и размах выборки.

Выборочная дисперсия s^2 – это сумма квадратов отклонений выборочных результатов наблюдений от их среднего арифметического, деленная на объем выборки:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2.$$

Выборочное среднее квадратическое отклонение s – квадратный корень из дисперсии:

$$s = \sqrt{s^2}.$$

На практике выборочное среднее квадратическое отклонение позволяет оценить, насколько значения из множества могут отличаться от среднего значения. Большее значение выборочного среднеквадратического отклонения показывает больший разброс значений в представленном множестве со средней величиной множества. Меньшее значение, соответственно, показывает, что значения в множестве сгруппированы вокруг среднего значения.

В данной работе под выборочным среднеквадратическим отклонением будем иметь ввиду риск попасть в аварию на определенном отрезке выбранного маршрута.

Глава 2. Метод анализа маршрута на примере

2.1. Анализ маршрута для водителя

Рассмотрим применение метода на конкретном примере. Будем строить маршрут от «Санкт-Петербург метро Елизаровская» до конечной точки «Санкт-Петербург метро Нарвская». Маршрут, построенный по этим данным, приведен ниже на рис. 4.

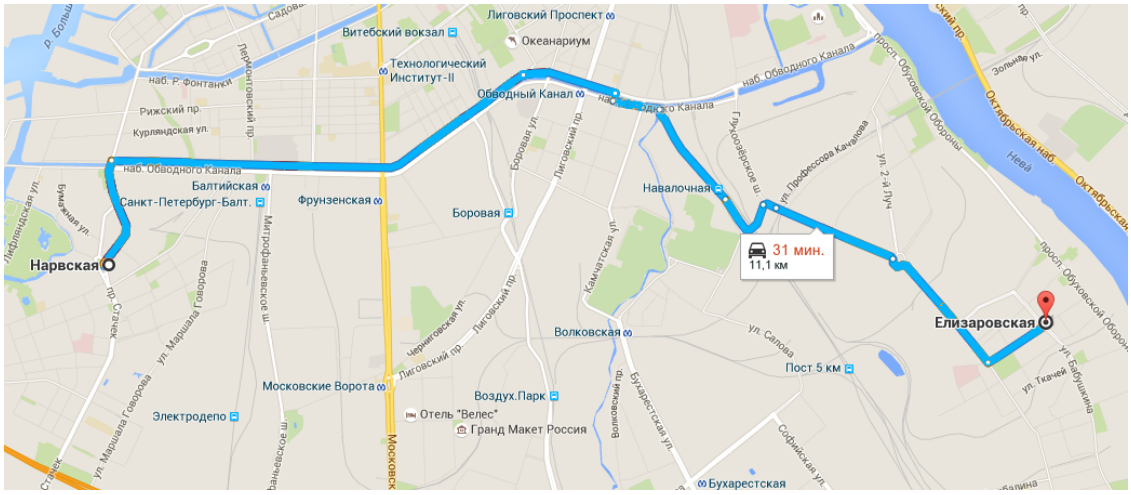

Рис. 4: Маршрут от м. Елизаровская до м. Нарвская

Далее разобьем маршрут на отрезки от перекрестка до перекрестка. В данной работе перекрестком будем считать смену одной улицы на другую. Разбиение на отрезки представлено на рис. 5.

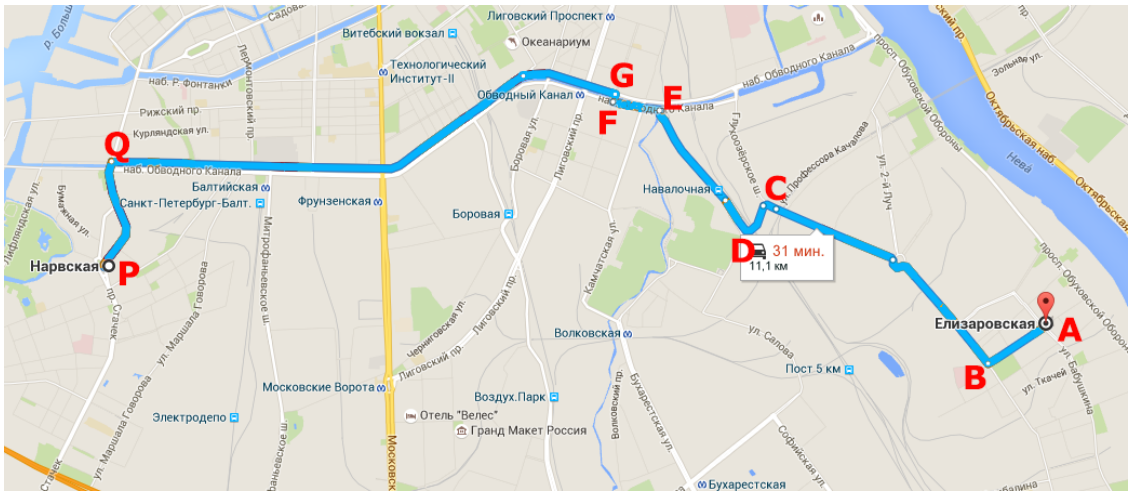

Рис. 5: Маршрут от м. Елизаровская до м. Нарвская

Для каждого построенного отрезка считаем количество аварий, которое в него входит в определенный день недели. Полученные расчеты представлены в таблице 1.

	AB	BC	CD	DE	EF	FG	GQ	QP
ПН	1	1	0	0	1	0	7	1
ВТ	0	1	0	0	0	0	7	0
СР	1	1	1	1	1	0	5	1
ЧТ	0	2	0	0	1	0	3	1
ПТ	0	0	0	0	1	0	3	1
СБ	0	0	0	0	1	0	8	1
ВС	0	1	0	0	1	0	8	0
Длинна отрезка (км)	0,60	2,50	0,30	1,30	0,70	0,13	4,60	1,20

Таблица 1: Количество аварий a_{ij}^d

Далее для каждого дня недели посчитаем долю аварий, которая приходится на каждый отрезок выбранного маршрута. Для этого делим результаты, полученные в первой таблице на общее количество аварий, которые произошли в определенный день недели. В таблице 2 представлены сведения об авариях в каждый день недели за 2015 год.

	Количество аварий
ПН	1040
ВТ	1057
СР	1007
ЧТ	1056
ПТ	1115
СБ	1083
ВС	885
Всего	7243

Таблица 2: Количество аварий A_{ij}^d

Посчитанные доли округленны до пяти знаков после запятой и представлены в таблице 3.

	AB	BC	CD	DE	EF	FG	GQ	QP
ПН	0,00096	0,00096	0	0	0,00096	0	0,00673	0,00096
ВТ	0,00000	0,00095	0	0	0,00000	0	0,00662	0,00000
СР	0,00099	0,00099	0	0	0,00099	0	0,00497	0,00099
ЧТ	0,00000	0,00189	0	0	0,00095	0	0,00284	0,00095
ПТ	0,00000	0,00000	0	0	0,00090	0	0,00269	0,00090
СБ	0,00000	0,00000	0	0	0,00092	0	0,00739	0,00092
ВС	0,00000	0,00113	0	0	0,00113	0	0,00904	0,00000

Таблица 3: Доли аварий на каждом отрезке (D_{ij}^d)

Используя полученные данные, найдем уровень аварийности, приходящуюся на 1 км за каждый день недели на каждом отрезке маршрута. Для этого поделим долю аварий на каждом отрезке на длину это отрезка. Расчеты приведены в таблице 4.

	AB	BC	CD	DE	EF	FG	GQ	QP
ПН	0,00160	0,00038	0	0	0,00137	0	0,00146	0,00080
ВТ	0,00000	0,00038	0	0	0,00000	0	0,00144	0,00000
СР	0,00166	0,00040	0	0	0,00142	0	0,00108	0,00083
ЧТ	0,00000	0,00076	0	0	0,00135	0	0,00062	0,00079
ПТ	0,00000	0,00000	0	0	0,00128	0	0,00058	0,00075
СБ	0,00000	0,00000	0	0	0,00132	0	0,00161	0,00077
ВС	0,00000	0,00045	0	0	0,00161	0	0,00197	0,00000

Таблица 4: x_{ij}^d - аварийность на 1 км отрезка k_{ij}

Далее найдем выборочное среднее значение уровня аварийности, приходящейся на 1 км отрезка. Тем самым мы найдем уровень аварийности, который приходится на 1 км всего маршрута. И уже используя данный показатель, можно будет делать выводы о построенном маршруте. Также найдем выборочное среднее квадратическое отклонение за каждый день недели для данного маршрута, тем самым мы найдем риски для этого маршрута. Вычисления представлены в таблице 5.

	Аварийность	Риски
ПН	0,00070	0,000655
ВТ	0,00023	0,000448
СР	0,00067	0,000591
ЧТ	0,00044	0,000455
ПТ	0,00033	0,000433
СБ	0,00046	0,000597
ВС	0,00050	0,000718

Таблица 5: Значения $\bar{x}_i^d$ и s_i^d

Т.к количество аварий в каждый день недели за 2015 год превышает число 1000, то значения уровня аварийности и рисков будут меньше чем 0,009. Поэтому для удобства умножим все значения в таблице 5 на 10^3 и получим окончательный результат.

	$\bar{x}_i^d * 10^{-3}$	$s_i^d * 10^{-3}$
ПН	0,7	0,655
ВТ	0,23	0,448
СР	0,67	0,591
ЧТ	0,44	0,455
ПТ	0,33	0,433
СБ	0,46	0,597
ВС	0,5	0,718

Таблица 6: Значения $\bar{x}_i^d * 10^{-3}$ и $s_i^d * 10^{-3}$

Для анализа было построено свыше 50 маршрутов и выявлено следующее правило:

1. Маршрут, построенный водителем, является безопасным, если выпол-

няется следующая система неравенств:

$$\begin{cases} 0 \leq \bar{x}_i^d * 10^{-3} \leq 0,37 \\ 0 \leq s_i^d * 10^{-3} \leq 0,465. \end{cases}$$

2. Маршрут, построенный водителем, относиться к средне безопасному если выполняется следующая система неравенств:

$$\begin{cases} 0,37 < \bar{x}_i^d * 10^{-3} \leq 0,63 \\ 0,465 < s_i^d * 10^{-3} \leq 0,623. \end{cases}$$

   Также маршрут является средне безопасным, если один из показателей относится к безопасному, а другой к средне безопасному.

3. Маршрут, построенный водителем, является опасным если выполняется хотя бы одно из неравенств:

$$\left[\begin{array}{l} \bar{x}_i^d * 10^{-3} > 0,63 \\ s_i^d * 10^{-3} > 0,623. \end{array} \right.$$

2.2. Анализ маршрута для пешехода

Рассмотрим данный метод анализа на маршруте построенном для пешехода. Начальная точка "Санкт-Петербург метро Автово конечная точка "Санкт-Петербург метро Кировский завод". Построенный и поделенный на участки маршурт представлен на рис. 6.

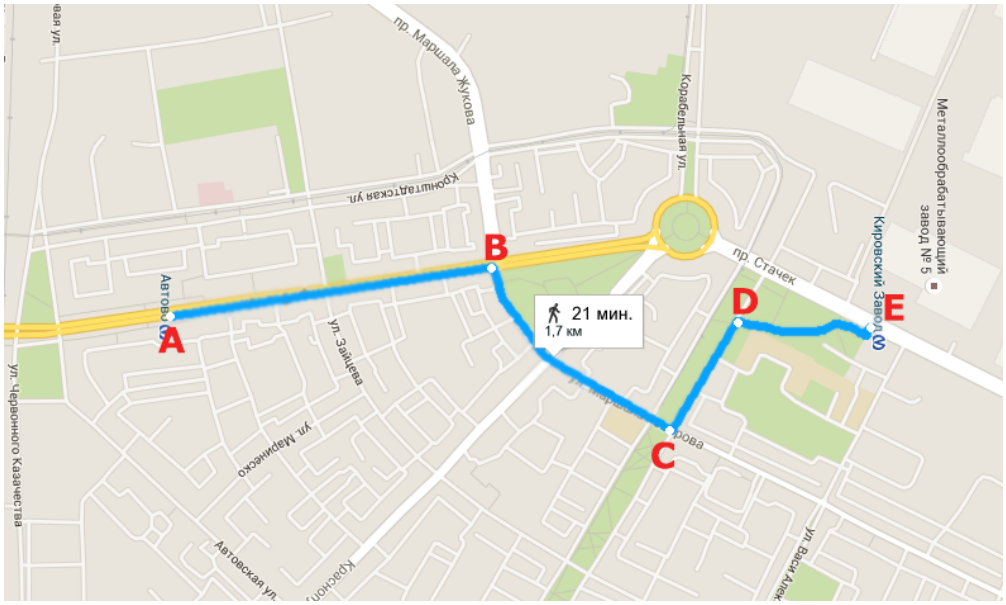

Рис. 6: Маршрут от м. Автово до м. Кировский завод

Все показатели маршрута для пешехода считаются аналогично показателям маршрута для водителя, за исключением того, что из базы берутся

только те аварии, которые относятся к виду "наезд на пешехода" и "падение пешехода". И уже из этих аварий выбираются те, которые произошли в данный день недели на улице, которой принадлежит участок маршрута.

Далее по тому же алгоритму, что и для водителя, считается количество аварий, которое произошло на конкретном участке. После подсчета количества аварий, осуществляется аналогичные вычисления и в конечном итоге по данному маршруту получаем следующий результаты:

	$\bar{x}_i^d * 10^{-3}$	$s_i^d * 10^{-3}$
ПН	2,40	4,15
ВТ	2,11	2,15
СР	0,88	1,53
ЧТ	2,71	2,83
ПТ	2,11	3,65
СБ	1,05	1,83
ВС	1,81	3,14

Таблица 7: Значения $\bar{x}_i^d * 10^{-3}$ и $s_i^d * 10^{-3}$

Показатели получились на порядок выше чем для водителя, т.к количество аварий вида "наезд на пешехода" и "падение пешехода"не превышает 500 за каждый день недели.

День недели (d)	Количество аварий (A^d)
ПН	417
ВТ	419
СР	435
ЧТ	468
ПТ	475
СБ	365
ВС	276

Таблица 8: Количество аварий с участием пешеходов

Т.к. пешеход намного меньше контактирует с автомобильными дорогами, то правило придуманное для классификации маршрутов, построенных водителем, не подойдет. Поэтому анализируя маршруты для пешехода было выдвинуто следующее правило:

1. Маршрут, построенный пешеходом, является безопасным, если выполняется следующая система неравенств:

$$\begin{cases} 0 \leq \bar{x}_i^d * 10^{-3} \leq 1,9 \\ 0 \leq s_i^d * 10^{-3} \leq 2,6. \end{cases}$$

2. Маршрут, построенный пешеходом, относиться к средне безопасному

если выполняется следующая система неравенств:

$$\begin{cases} 1,9 < \bar{x}_i^d * 10^{-3} \leq 2,64 \\ 2,6 < s_i^d * 10^{-3} \leq 3,7. \end{cases}$$

Также маршрут является средне безопасным, если один из показателей относится к безопасному, а другой к средне безопасному.

3. Маршрут, построенный пешеходом, является опасным если выполняется хотя бы одно из неравенств:

$$\begin{bmatrix} \bar{x}_i^d * 10^{-3} > 2,64 \\ s_i^d * 10^{-3} > 3,7. \end{bmatrix}$$

В дальнейшем используя показатели по каждому построенному маршруту, можно будет произвести кластерный анализ, и реализовать программно самообновляющуюся классификацию аварий. Т.е. в зависимости от того сколько человек воспользовалось сервисом программа будет производить анализ показателей по этим маршрутам и изменять правило классификации. Но для реализации необходимо чтобы изначально в базе построенных маршрутов было как минимум свыше 500 маршрутов, для наиболее точной классификации.

Глава 3. Программная реализация

3.1. Сбор данных

Данные об авариях на дорогах Санкт-Петербурга были собранны с официального сайта ГИБДД. С сайта информация об авариях качается в формате xls файлов. Далее программа разбирает скаченный файл и забирает оттуда нужную информацию. Была предпринята попытка полностью автоматизировать процесс обновления базы, но удалось реализовать только некоторые этапы автоматизации:

1. Автоматически происходит разбор скаченного файла;
2. Автоматически происходит обновление базы с добавлением в нее новых аварий.

Не удалось реализовать автоматическое скачивание файлов с сайта ГИБДД, так как на сайте стоит ограничение по количеству запросов в минуту с одного компьютера. И в случае превышения лимита блокируется возможность обращаться на сайт. В данный момент на сайте висит объявление о том, что вскоре будет разработан GIBDD API, с помощью которого можно будет получать информацию об авариях в автоматизированном режиме. И тем самым можно будет расширить сервис по всем регионам России.

После сбора происходила обработка данных, и все данные были внесены в одну большую базу. Для хранения данных была выбрана база MySQL [7]. О каждой аварии хранится следующая информация: универсальный номер (id) каждой аварии, день, в который произошла авария, район аварии, вид дтп, точный адрес ближайшего дома, рядом с которым произошла авария. Фрагмент собранной базы представлен на рис. 7.

avariya_id	avariya_data	avariya_area	avariya_view	avariya_addres
1	2015-12-29	Петродворцовый район	Наезд на пешехода	Петродворцовый район, п Стрельна, ш Санкт-Петербургское, 46
2	2015-12-28	Петродворцовый район	Столкновение	Петродворцовый район, п Стрельна, ш Санкт-Петербургское, 76
3	2015-12-26	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ш Ропшинское, 4
4	2015-12-22	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ул Аврова, 45
5	2015-12-20	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ш Ропшинское, 4
6	2015-12-18	Петродворцовый район	Падение пассажира	Петродворцовый район, г Ломоносов, ул Федюнинского, 16
7	2015-12-17	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ул Александровская, 20-16
8	2015-12-16	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ш Гостилицкое, 13-1
9	2015-12-15	Петродворцовый район	Столкновение	Петродворцовый район, п Стрельна, 78
10	2015-12-14	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ул Михайловская, 5
11	2015-12-14	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Ломоносов, пр-кт Ораниенбаумский, 31
12	2015-12-14	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, б-р Разведчика, 2 К.1
13	2015-12-14	Петродворцовый район	Наезд на препятствие	Петродворцовый район, г Петергоф, ул Гостилицкая, 131
14	2015-12-13	Петродворцовый район	Столкновение	Петродворцовый район, г Петергоф, ш Санкт-Петербургское, 97
15	2015-12-13	Петродворцовый район	Наезд на препятствие	Петродворцовый район, г Санкт-Петербург, ш Санкт-Петербургское, 109
16	2015-12-10	Петродворцовый район	Столкновение	Петродворцовый район, г Петергоф, ул Астрономическая, 6
17	2015-12-10	Петродворцовый район	Наезд на стоящее ТС	Петродворцовый район, г Петергоф, ш Ропшинское, 1км
18	2015-12-08	Петродворцовый район	Наезд на пешехода	Петродворцовый район, г Петергоф, ул Разводная, 11-50
19	2015-12-07	Петродворцовый район	Наезд на пешехода	Петродворцовый район, п Стрельна, ш Санкт-Петербургское, 115
20	2015-12-05	Петродворцовый район	Столкновение	Петродворцовый район, п Стрельна, ш Санкт-Петербургское, 63
21	2015-11-26	Петродворцовый район	Столкновение	Петродворцовый район, г Петергоф, ул Гостилицкая, 28
22	2015-11-25	Петродворцовый район	Наезд на препятствие	Петродворцовый район, г Ломоносов, ул Морская, 94

Рис. 7: Фрагмент базы с авариями за 2015 год

Для хранения информации о маршруте была создана дополнительная база, куда записывается информация о каждом построенном маршруте пользователя, а именно: точка отправления, точка прибытия, общая длина маршрута, аварийность маршрута, приходящаяся на 1 км, и выборочное среднеквадратическое отклонение (риск маршрута).

3.2. Построение маршрута

При построении маршрута пользователь видит только карту, на которой изображен маршрут, как представлено на рис. 4. Программа в этот момент получает все координаты перекрестков, через которые проедет пользователь, и делит маршрут на отрезки от перекрестка до перекрестка. Далее попарно берутся координаты перекрестков, которые являются концами отрезков, и строится прямоугольник следующего вида (см. рис. 8):

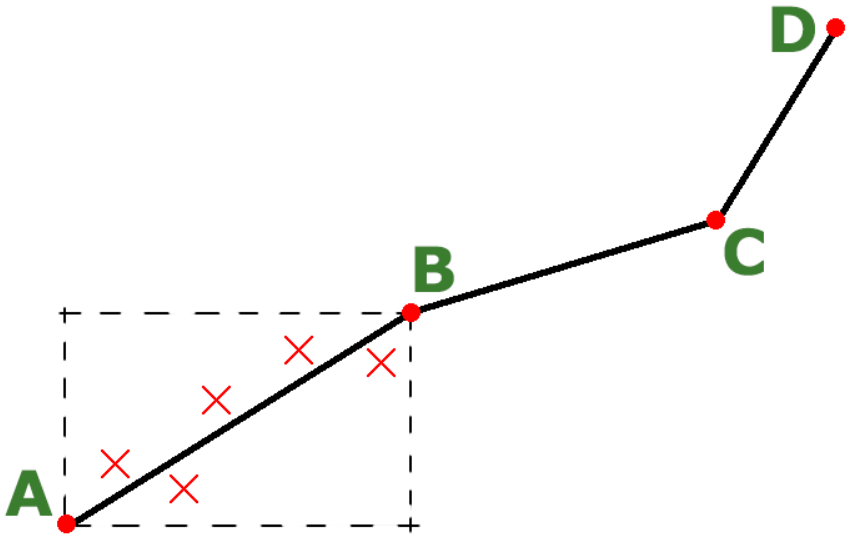

Рис. 8: Количество аварий на отрезке AB

Далее из базы запрашиваются все аварии, которые произошли на улице AB в данный день недели. И уже из этих аварий выбираются только те, которые попадают в прямоугольник, представленный на рис. 8. Алгоритм проверки следующий:

1. Берутся адреса, запрошенных ранее аварий (которые произошли в данный день недели и на улице, которой принадлежит конкретный отрезок);
2. С помощью службы геокодирования эти адреса переводятся в координаты;
3. На этом этапе происходит проверка, проверяем, какие из координат входят в прямоугольник, который показан на рис. 8.

Разберем третий этап подробнее. Пусть у точки A координаты (x_a, y_a), у точки $B(x_b, y_b)$. И есть некоторая авария - назовем ее $T(x_t, y_t)$. Она произошла в данный день недели на улице, которой принадлежит наш отрезок AB. Далее происходит проверка на истинность следующих неравенств:

$$\begin{cases} min(x_a, x_b) < x_t < max(x_a, x_b) \\ min(y_a, y_b) < y_t < max(y_a, y_b) \end{cases}$$

Если оба неравенства выполняются, то координаты $T(x_t, y_t)$ входят в построенный прямоугольник, и мы относим эту аварию к отрезку AB. Затем берем следующую аварию и также проверяем.

Такая процедура проделывается для каждого отрезка. В результате получаем количество аварий на каждом отрезке в определенный день недели.

3.3. Анализ построенного маршрута

После подсчета количества аварий на каждом отрезке построенного маршрута, начинается расчет аварийности и рисков. Алгоритм анализа следующий:

1. Подсчитанное количество аварий на каждом отрезке делится на общее количество аварий в данный день недели. Тем самым мы находим долю аварийности, приходящуюся на данный отрезок пути (D_{ij}^d);
2. Полученные данные из пункта один делим на длину каждого отрезка и находим долю аварийности, которая приходится на 1 км отрезка (x_{ij}^d);
3. Считаем среднюю аварийность на 1 км по всем отрезкам и получаем аварийность, которая приходится на 1 км всего маршрута ($\bar{x}_{ij}^d$);
4. Далее находим выборочное среднее квадратическое отклонение по аварийности на 1 км по каждому отрезку и получаем риск попасть в аварию по данному маршруту (s_{ij}^d);
5. При анализе полученных данных из пункта 3 и 4 происходит классификация маршрута на опасный и безопасный. Также при анализе данных из пункта 2 выявляются наиболее опасные участки маршрута.

В случае, когда Google API предложит несколько вариантов маршрутов, то будет произведен аналогичный анализ для каждого маршрута. Рассмотрим случай когда будет возвращено 3 варианта маршрута.

Разберем подробнее то, что изображено на рис. 9. Из пункта A в B можно попасть тремя способами, первый маршрут $K_1 = (k_{11})$ состоит только из одного отрезка. Второй маршрут состоит из трех отрезков $K_1 = (k_{21}, k_{22}, k_{23})$ и соответственно третий маршрут из четырех отрезков

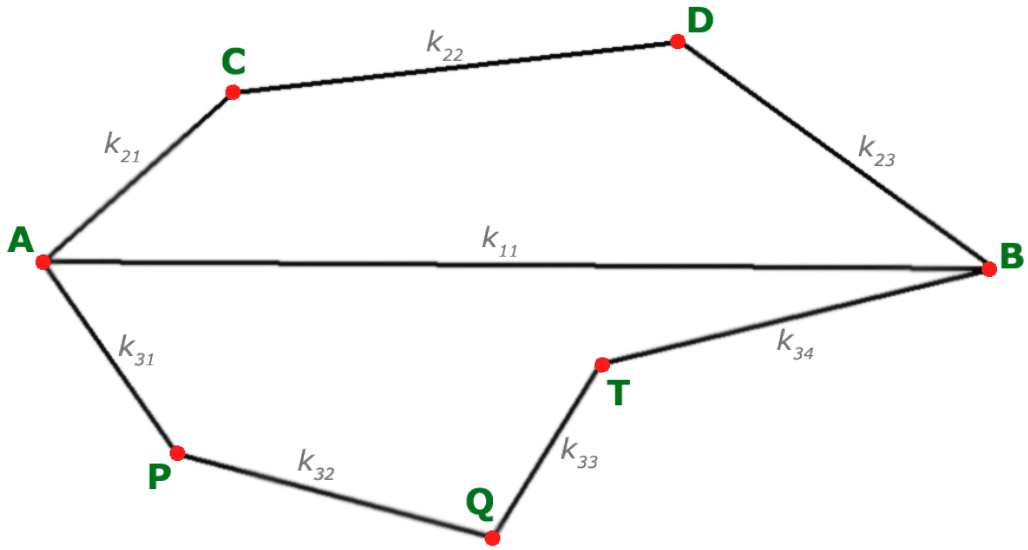

Рис. 9: Варианты маршрутов из токи A в B

$K_3 = (k_{31}, k_{32}, k_{33}, k_{34})$. Каждый маршрут проходит по алгоритму анализа, который представлен выше. И в результате расчетов программа получает следующую информацию:

d	$\bar{x}_1^d * 10^{-3}$	$s_1^d * 10^{-3}$	$\bar{x}_2^d * 10^{-3}$	$s_2^d * 10^{-3}$	$\bar{x}_3^d * 10^{-3}$	$s_3^d * 10^{-3}$
0	$\bar{x}_1^0$	s_2^0	$\bar{x}_2^0$	s_2^0	$\bar{x}_3^0$	s_3^0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
6	$\bar{x}_1^6$	s_2^6	$\bar{x}_2^6$	s_2^6	$\bar{x}_3^6$	s_3^6

Таблица 9: Информация по каждому маршруту (K_1, K_2, K_3)

Где $\bar{x}_i^d$ средняя аварийность i-го маршрута на 1 км, а s_i^d риск i-го маршрута в d-ый день недели.

Таким образом, исходя из того какой сегодня день недели, другими словами чему равно d, пользователю будет предложен наименее опасный маршрут, тот маршрут у которого наименьшие показатели $\bar{x}_i^d$ и s_i^d в d-ый день недели.

Пример работы этого алгоритма представлен в гл. 2 "Метод анализа маршрута на примере".

Выводы

Таким образом, анализируя каждый построенный маршрут пользователя, мы сообщаем ему не только стандартную информацию, но и информируем его об опасности построенного маршрута, выявляя опасные участки дорог и предлагая ему по возможности их объехать.

Возвращаясь к статистике о том, что в среднем в Санкт-Петербурге за 2015 год в день происходило около 20-ти аварий, хотим отметить, что если данный сервис повысит внимательность водителей и пешеходов и снизит этот показатель до 19, то это будет очень хорошим результатом.

Также хорошим результатом будет, если данные исследования, методы и реализованные технологии будут использоваться в дальнейшем для обеспечения безопасности на дорогах.

Заключение

В результате проделанной работы удалось создать общедоступный сервис [11], который на основе статистических данных об аварийности в Санкт-Петербурге за 2015 год, обрабатывает и анализирует произвольный маршрут, заданный пользователем (водителем или пешеходом). Выдает информацию и дает рекомендации объехать или обойти наиболее опасные участки маршрута.

В работе используются данные об аварийности в городе Санкт - Петербург за 2015 год, но как отмечалось выше, программный продукт собран таким образом, чтобы с минимальными правками расширить работу сервиса для всех регионов России. Проблема состоит только в автоматизации получения данных об аварийности.

Также в данной работе маршрут классифицируется как опасный или безопасный по правилу, которое было выявлено опытным путем, было построено некоторое количество маршрут и, исходя из этих показателей, было выдвинуто правило. Но в случае, когда сервисом воспользуются свыше 500 человек, можно будет обновить сервис, добавив в него кластерный анализ [8]. Таким образом, чтобы после каждого нового маршрута правило классификации пересчитывалось, и, тем самым, можно добиться более точной классификации.

Список литературы

[1] Официальный сайт ГИБДД. http://www.gibdd.ru.

[2] Орлов А.И., Математика случая: Вероятность и статистика – основные факты, Учебное пособие. – М.: МЗ-Пресс, 2004. 176 с.

[3] Буре В. М., Парилина Е. М. Теория вероятностей и математическая статистика, издательство "Лань" , 2013. 416 с.

[4] Форсье Д., Биссекс П., Чан У., Разработка веб-приложений, издательство Символ-Плюс, 2010. 456 с.

[5] Google Maps API. https://developers.google.com/maps/?hl=ru.

[6] Google Maps Directions API. https://developers.google.com/maps/documentation/directions/?hl=ru.

[7] Робин Никсон, Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript, 2011. 418 с.

[8] Дюрбан Б., Оделл П., Кластерный анализ, Москва "Статистика" , 1977. 128 с.

[9] Бен Хеник, HTML и CSS путь к совершенству, 2011. 240 с.

[10] Капский Д. В., Пегин П. А. Методика прогнозирования аварийности на регулируемом перекрестке, журнал "Наука и техника" , выпуск № 5, 2015.

[11] Сервис представленный в данной работе. http://avariyamnet.ru.

Приложение

models.py

```
1
2 # -*- coding: utf-8 -*-
3 from __future__ import unicode_literals
4 from django.db import models
5
6 # Create your models here.
7
8 class Avarii(models.Model):
9     class Meta():
10         db_table = 'avarii'
11     avariya_id = models.IntegerField()
12     avariya_data = models.DateField()
13     avariya_area = models.TextField()
14     avariya_view = models.CharField(max_length=100)
15     avariya_addres = models.CharField(max_length=200)
16     avariya_death = models.IntegerField()
17     avariya_woudn = models.IntegerField()
18     avariya_cars = models.IntegerField()
19     avariya_people = models.IntegerField()
```

views.py

```
1 # -*- coding: utf-8 -*-
2 from django.shortcuts import render_to_response
3 from models import Avarii
4 import datetime
5 # Create your views here.
6
7 def statistic_on_request(request):
8     death=0
9     car=0
10    people=0
11    wound=0
12    date=[]
13    addres=[]
14    number=[]
15    num=[]
16    file = Avarii.objects.filter(avariya_view=request)
17    all = len(file)
18    for key in file:
19        if datetime.datetime.strftime(key.avariya_data, "%d.%m.%y") not in
              date:
20            date.append(datetime.datetime.strftime(key.avariya_data, "%d.%m.%
                  y"))
```

```
            number.append(len(Avarii.objects.filter(avariya_data=key.
                avariya_data)))
    for i in Avarii.objects.filter(avariya_area=request):
        addres.append(i.avariya_addres)
    for k in addres:
        num.append(addres.count(k))
    for address in file:
        if datetime.datetime.weekday(address.avariya_data) == 6:
            date.append(address.avariya_addres)
    for i in file:
        death += i.avariya_death
        car += i.avariya_cars
        people += i.avariya_people
        wound += i.avariya_woudn

    return render_to_response('main.html',{'death': death,'all': all,'car':
        car,'people': people, 'woudn': wound, 'a':addres, 'date':date, '
        number':len(date)})
```

settings.py

```
"""
Django settings for niralbert project.

Generated by 'django-admin startproject' using Django 1.9.1.

For more information on this file, see
https://docs.djangoproject.com/en/1.9/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.9/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.9/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '*ut!bb$==hd^#^#@!-q1%0w2(0y1rcv7urr1!o3cl50lo$3d$5'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []
```

```
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'update',
41     'planning_route_and_calculat',
42 ]
43
44 MIDDLEWARE_CLASSES = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
51     'django.contrib.messages.middleware.MessageMiddleware',
52     'django.middleware.clickjacking.XFrameOptionsMiddleware',
53 ]
54
55 ROOT_URLCONF = 'niralbert.urls'
56
57 TEMPLATES = [
58     {
59         'BACKEND': 'django.template.backends.django.DjangoTemplates',
60         'DIRS': ['/Users/ht_albert/djangoenv/bin/niralbert/templates/',],
61         'APP_DIRS': True,
62         'OPTIONS': {
63             'context_processors': [
64                 'django.template.context_processors.debug',
65                 'django.template.context_processors.request',
66                 'django.contrib.auth.context_processors.auth',
67                 'django.contrib.messages.context_processors.messages',
68             ],
69         },
70     },
71 ]
72
73 WSGI_APPLICATION = 'niralbert.wsgi.application'
74
75
76 # Database
77 # https://docs.djangoproject.com/en/1.9/ref/settings/#databases
```

```
78
79 DATABASES = {
80     'default': {
81         'ENGINE': 'django.db.backends.sqlite3',
82         'NAME': os.path.join(BASE_DIR, 'db_niralbert'),
83     }
84 }
85
86
87 # Password validation
88 # https://docs.djangoproject.com/en/1.9/ref/settings/#auth-password-
       validators
89
90 AUTH_PASSWORD_VALIDATORS = [
91     {
92         'NAME': 'django.contrib.auth.password_validation.
              UserAttributeSimilarityValidator',
93     },
94     {
95         'NAME': 'django.contrib.auth.password_validation.
              MinimumLengthValidator',
96     },
97     {
98         'NAME': 'django.contrib.auth.password_validation.
              CommonPasswordValidator',
99     },
100    {
101        'NAME': 'django.contrib.auth.password_validation.
              NumericPasswordValidator',
102    },
103 ]
104
105
106 # Internationalization
107 # https://docs.djangoproject.com/en/1.9/topics/i18n/
108
109 LANGUAGE_CODE = 'en-us'
110
111 TIME_ZONE = 'UTC'
112
113 USE_I18N = True
114
115 USE_L10N = True
116
117 USE_TZ = True
118
119
120 # Static files (CSS, JavaScript, Images)
121 # https://docs.djangoproject.com/en/1.9/howto/static-files/
```

```
122
123 STATIC_URL = '/static/'
124
125 STATICFILES_DIRS = [
126     ('static', '/Users/ht_albert/djangoenv/bin/niralbert/static'),
127 ]
```

views–post.py

```
1  # -*- coding: utf-8 -*-
2
3  from django.shortcuts import render
4  from django.http.response import HttpResponse, Http404
5  from django.template.loader import get_template
6  from django.template import Context
7  from django.shortcuts import render_to_response, redirect
8  from article.models import Article, Comments
9  from setting.models import Settings
10 from django.core.exceptions import ObjectDoesNotExist
11 from forms import CommentForm
12 from django.core.context_processors import csrf
13 from django.contrib import auth
14 from django.core.paginator import Paginator
15 # Create your views here.
16
17
18 #Step_3
19
20
21 def basic_one(request):
22     t = get_template("myview.html");
23     html = t.render(Context({'name': view}))
24     return HttpResponse(html)
25
26
27 def basic_two(request):
28     return render_to_response('myview.html',{'name': view} )
29
30
31 def articles(request, page_number=1):
32     all_articles = Article.objects.all()
33     settings = Settings.objects.get(id=1)
34     current_page = Paginator(all_articles, settings.blog_post)
35     return render_to_response('articles.html',{'articles': current_page.page
           (page_number),
36                                                'username': auth.get_user(request
                                                      ).username,
37                                                'settings': settings.blog_name,
38                                                },
```

```
                                )


def article(request, article_id=1):
    comment_form = CommentForm
    args = {}
    args.update(csrf(request))
    args['article'] = Article.objects.get(id=article_id)
    args['comments'] = Comments.objects.filter(comments_article_id=
        article_id)
    args['form'] = comment_form
    args['username'] = auth.get_user(request).username
    return render_to_response('article.html', args)



def addlike (request, article_id):
    try:
        if article_id in request.COOKIES:
            pass
        else:
            article = Article.objects.get(id=article_id)
            article.article_like += 1
            article.save()
            response = redirect('/')
            response.set_cookie(article_id, "test")
            return response
    except ObjectDoesNotExist:
        raise Http404
    return redirect('/')



def addcomment (request, article_id):
    if request.POST and ("pause" not in request.session): #если поучили
        данные и нет сессии с именем пауза
        form = CommentForm(request.POST)
        if form.is_valid(): #валидация формы
            comment = form.save(commit=False)
            comment.comments_article = Article.objects.get(id=article_id)
            form.save()
            request.session.set_expiry(60)
            request.session['pause'] = True
    return redirect("/articles/get/%s" % article_id)

def add(request):
    return render_to_response('main.html', {'null': Article.article_like.
        filter(id=1)},)
```